Semantics and Fitness Landscapes in Genetic Programming

Leonardo Vanneschi
lvanneschi@isegi.unl.pt
Objective

Presenting new operators for Genetic Programming

Key concepts:

• Geometry
• Semantics
• Fitness Landscapes
Before beginning some excellent news…

FCT Project PTDC/EEI-CTP/2975/2012:
*Improving Semantic Genetic Programming for Maritime Safety, Security and Environmental Protection*

Recommended for Funding!!

The Team:

Leonardo Vanneschi, ISEGI, Principal Investigator

Mauro Castelli
ISEGI

Sara Silva
INESC-ID

Ernesto Costa
Univ. Coimbra

Ricardo Maia
Critical Software

Alberto Moraglio
Univ. Birmingham
Even better news…

Grants !!!!!!
Agenda

- Optimization Problems
- Fitness Landscapes
- Genetic Algorithms (geometric operators)
- Genetic Programming (geometric semantic operators)
- Our implementation of geometric semantic operators
- Discussion
- Open issues
Optimization Problems

Informally

solving an optimization problem means to find the best solution(s) in a (typically huge) set of other candidate solutions

A little bit more formally

A pair: \((S, f)\)

where \(S\) is the set of all possible solutions (we will call it search space) and \(f\) is a function:

\[ f = S \rightarrow R \]

\(f\) quantifies the quality of the solutions in \(S\) and it is called cost function or fitness function
Optimization Problems - Definitions

An optimization problem is a *minimization problem* if it consists in looking for a solution \( o \in S \), such that:

\[
f(o) \leq f(i), \ \forall \ i \in S
\]

An optimization problem is a *maximization problem* if it consists in looking for a solution \( o \in S \), such that:

\[
f(o) \geq f(i), \ \forall \ i \in S
\]

Such a solution is called *global optimum* (minimum or maximum)
Hill Climbing

The most natural and immediate method to solve an optimization problem.

It consists in trying to improve fitness step by step (stepwise improvement) by means of the concept of neighborhood.

\[
\text{initialize } (i_{\text{start}}); \\
i := i_{\text{start}}; \\
\text{repeat} \\
\quad \text{GENERATE (j from } N_i); \\
\quad \text{if } f(j) \text{ better than } f(i) \text{ then } i := j; \\
\text{until } f(i) \text{ better than } f(j), \ \forall j \in N_i;
\]

\(N_i = \text{neighborhood of solution } i\)
Hill Climbing - Example

Consider the following maximization problem:

\[ S = \{ i \mid i \in \mathbb{N} \text{ and } 0 \leq i \leq 15 \} \]

\[ \forall i \in S, \ f(i) = \text{number of "1"s in the binary representation of } i \]

Neighborhood: \[ j \in N_i \iff |j - i| = 1 \]

Steps of the algorithm:

• Current solution (randomly generated): \( i = 6 \ (0110) \) \( f(i) = 2 \)
  neighbors of \( i = 5 \ (0101), 7 \ (0111) \) \( f(5) = 2, f(7) = 3 \) new current solution: \( i := 7 \)

• neighbors of \( i = 6 \ (0110), 8 \ (1000) \) \( f(6) = 2, f(8) = 1 \) algorithm terminates

Solution returned: \( i = 7 \ (0111) \) \( f(i) = 3 \) \( \text{The solution returned by the Hill Climbing is a "local optimum".} \)

Global optimum: \( o = 15 \ (1111) \) \( f(o) = 4 \)
Local Optima

A solution \( j \in S \) is called \textbf{local optimum (as regards a neighborhood structure} \( N \)), if:

\[
\begin{align*}
\text{for minimization problems:} & \quad f(j) \leq f(i) \quad \forall \; i \in N_j \\
\text{for maximization problems:} & \quad f(j) \geq f(i) \quad \forall \; i \in N_j
\end{align*}
\]

Hill Climbing \textbf{always} returns a local optimum \textit{(not necessarily corresponding to the global one)}.
Fitness Landscape

...So, let us consider again the previous example and let us draw a plot: horizontal axis: all the solutions in the search space (ordered according to the neighborhood structure); vertical axis: fitness.

Hill Climbing
Fitness Landscape

Fitness landscape \((S, \mathcal{V}, f)\):

- \(S\) : set of potential solutions,
- \(\mathcal{V} : S \rightarrow 2^S\) : neighborhood function,
- \(f : S \rightarrow \mathbb{R}\) : fitness function.

\(\mathcal{V} : S \rightarrow 2^S\) : neighborhood function
\[\forall x \in S,\]

\[\mathcal{V}(x) = \{y \in S \mid y = op(x)\}\]

\[\mathcal{V}(x) = \{y \in S \mid d(y, x) \leq 1\}\]
Importance of Fitness Landscape

It gives a *visual intuition* of the *facility* or *difficulty* of a search agent (like Hill Climbing, but also Evolutionary Algorithms) to find the global optimum. For instance:

- Smooth landscape, with only one "peak" (global optimum) → easy problem
- Rugged landscape, with many local optima → hard problem

Limitation of fitness landscapes

It is generally *impossible to draw* a fitness landscape:

- Huge search space
- Huge neighborhoods (*multi-dimensionality!*)
Remark that...

if we consider exactly the same problem, but with a **different neighborhood structure**, the fitness landscape changes and Hill Climbing easily finds easily the global optimum:

\[
S = \{ i \mid i \in \mathbb{N} \land 0 \leq i \leq 15 \}
\]

\[\forall i \in S, \quad f(i) = \text{number of "1"s in the binary representation of } i\]

Neighborhood: \( j \in N_i \iff j \text{ and } i \text{ differ by just 1 bit} \)

There are **no local optima** in this fitness landscape! **Unimodal fitness landscape**.

(every individual that is different from the global optimum has at least one neighbor better than him, that can be obtained by changing a 0 into a 1).
Another Case

$S = \{ \text{vectors of prefixed length of real numbers included in } [0,10] \}$

$\forall i \in S, \quad f(i) = \text{distance to a prefixed (and known and unique) global optimum}$

Neighborhood: $j \in N_i \iff j \text{ is equal to } i \text{ except for the random } \textit{perturbation}$

of some of its coordinates of a quantity included in $[0,1]$.

Example

The global optimum $[8.0, 6.0, 4.0, 7.0, 5.0]$

A solution $i$ $[5.2, 6.4, 2.1, 4.9, 3.7]$

A solution $j$ neighbor of $i$ $[5.8, 6.4, 2.9, 4.9, 3.6]$

$closer!$
Terminology

The operator that is “related” to the neighborhood structure of the previous example is called **ball mutation**.

Let us give a name to the previous problem: **Distance Optimization with Known Optimum (DOKO)**
Overcome limitations Hill Climbing?

• **Simulated Annealing**
  it allows to "go down a hill" with a certain probability

• **Evolutionary Algorithms**
  • they do not consider just one "agent" but a population.
  • They try to improve the quality of the solutions imitating the Darwin’s theory of evolution (Darwin, 1859).
  • Reproduction
  • Ability of adapting to the surrounding environment
  • Heritability
  • Variation
  • Competition
The Evolutionary Process

Initial Population → Selection → Intermediary Population

... and the winner is...

Genetic Operators (Variation)

New Population
Genetic Algorithms (GAs)

Solutions/Individuals = *Strings of prefixed length*

“Traditional situation”: the values in each allele are discrete and:

1 0 1 1 0 0
0 0 1 0 1 1

---

1 0 1 1 1 1
0 0 1 0 0 0

---

1 1 0 0 1 0

---

1 1 1 0 1 0

---

1 1 1 0 1 0
Genetic Algorithms (GAs)

... but GAs can work also on vectors of *continuous* values.

In that case, many operators have been introduced.

Interesting: **Geometric operators** [Moraglio and Poli, 2006]:

**Geometric Crossover**

Coordinates of the (unique) offspring are the weighted average of the corresponding coordinates of the parents (with weighs in [0,1] whose sum is 1).

![Diagram showing Geometric Crossover]

- **Global optimum**
- **the offspring cannot be worse than the worse of the parents**
- this crossover expresses “betweennes”: the offspring stands between (in the line joining) the parents
Genetic Algorithms (GAs)

... but GAs can work also on vectors of continuous values.

In that case, many operators have been introduced.

Interesting: Geometric operators [Moraglio and Poli, 2006]:

Geometric Mutation

Ball mutation

Unimodal fitness landscape on the DOKO problem.
Genetic Programming (GP)

An evolutionary algorithm in which solutions/individuals are computer programs. Typically ("Lisp-like") trees.

Example

Given the set of data:

\[
H = \begin{pmatrix}
 2 & 4 & 10 \\
 3 & 5 & 13 \\
\end{pmatrix}
\]

A possible individual is:

\[
P(x_1, x_2) = x_2 \times (x_1 + x_2)
\]

It represents the program/function/expression: \( P(x_1, x_2) = x_2 \times (x_1 + x_2) \)

And one possible fitness could be:

\[
fitness(P) = |P(2,4) - 10| + |P(3,5) - 13| =
\]

\[
|4 \times (2+4) - 10| + |5 \times (3+5) - 13| = |24 - 10| + |40 - 13| = 14 + 27 = 41
\]
GP as a Machine Learning Method

- **Known**: the correct outputs for a fixed given set of inputs \( \{I_i, O_i\} \)

- **Sought**: a function belonging to a certain class that interpolates those points, i.e., \( f(I_i) = O_i \) for any \( i \)

- **Output vector**: the vector of the outputs of \( f \) is \( f(I) = (f(I_i)) \)

- **Fitness**: a measure on the error on the training set, i.e., *distance* between the output vectors of \( f \) and the target output vector \( F(f) = D(f(I), O) \) (ERROR AS DISTANCE)
What is «Semantics»?

Many definition exist, but in GP the most used one is:

*The vector of outputs of a program on the different training data*

**Example**

Given the set of data:  

\[ H = \begin{pmatrix} 2 & 4 & 10 \\ 3 & 5 & 13 \end{pmatrix} \]

The individual:

That represents function: \( P(x_1, x_2) = x_2 \times (x_1 + x_2) \)

has a semantics qual to:  

\([P(2,4), P(3,5)] = [24, 40]\)
GP as a Machine Learning Method

- Known: the correct outputs for a fixed given set of inputs \( \{I_i, O_i\} \)

- Sought: a function belonging to a certain class that interpolates those points, i.e., \( f(I_i) = O_i \) for any \( i \)

- Output vector: the vector of the outputs of \( f \) is \( f(I) = (f(I_i)) \)

- Fitness: a measure of the error on the training set, i.e., distance between the output vector of \( f \) and the target output vector
  \( F(f) = D(f(I), O) \) (ERROR AS DISTANCE)

Leonardo Vanneschi  Semantics and Fitness Landscapes in Genetic Programming
«Traditional» GP operators

crossover

mutation
«Traditional» GP operators

... Produce offspring by blind *syntactic manipulation* of parent parse trees, regardless of their semantics.

... Preserve syntactic “genetic material”, but what is their *effect on semantics*?
Objective

Is it possible to define transformations on the syntax of individuals that have known effects on their semantics?

For instance

Let us assume that we are able to find a transformation on the syntax of an individual whose effect is *ball mutation on the semantic space*. This transformation, if known, would induce a *unimodal fitness landscape on every problem* consisting in matching input data into known targets (e.g. regressions and classifications),

GP should have a good evolvability on those problems, at least on training data (we would *map the problem into the DOKO*).

Analogous observations hold for transformations on pairs of solutions that correspond to GA semantic crossovers.
Is it a dream?

Yes... but turning to reality

Even though with a big drawback (discussed later) those operators have been defined:

Geometric Semantic Crossover  [Moraglio et al., 2012]

**Definition 1. (Geometric Semantic Crossover).** Given two parent functions \( T_1, T_2 : \mathbb{R}^n \rightarrow \mathbb{R} \), the geometric semantic crossover returns the real function \( T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2) \), where \( T_R \) is a random real function whose output values range in the interval \([0, 1]\).

\[
T_{XO} = \begin{array}{c}
+ \\
/ \\
/ \\
T_1 \\
* \\
/ \\
1 \\
/ \ \\
T_R
\end{array} \\
\begin{array}{c}
* \\
/ \\
T_R \\
/ \\
T_2
\end{array}
\]

\( T_R = \) Random function with codomain \([0, 1]\)
Geometric Semantic Crossover  [Moraglio et al., 2012]

The (only) offspring generated by this *crossover has a semantic vector* that is a *linear combination of the semantics* of the parents with random coefficients included in $[0,1]$ and whose sum is equal to 1.
Geometric Semantic Mutation  [Moraglio et al., 2012]

Definition 2. (Geometric Semantic Mutation). Given a parent function \( T : \mathbb{R}^n \rightarrow \mathbb{R} \), the geometric semantic mutation with mutation step \( ms \) returns the real function \( T_M = T + ms \cdot (T_{R1} - T_{R2}) \), where \( T_{R1} \) and \( T_{R2} \) are random real functions.

\[
T_M = T + ms \cdot (T_{R1} - T_{R2})
\]

\( T_{R1}, T_{R2} = \) Random functions
Geometric Semantic Mutation  [Moraglio et al., 2012]

Each element of the semantic vector of the offspring is a “weak” perturbation of the corresponding element in the parent’s semantics.

“Weak” because \((T_{R1} - T_{R2})\) is centered in zero

It’s importance can be tuned by \(ms\).

It corresponds to ball mutation in the semantic space, so the fitness landscape it induces is unimodal (we match each problem consisting in matching inputs into targets into the DOKO!).
Geometric Semantic Operators

Moraglio et al. show interesting results on a set of benchmarks, but....

\[ XO(T_1, T_2) = (T_1 \cdot TR) + ((1 - TR) \cdot T_2) \]

At generation 2 (if we use only crossover), all the trees have this shape, so to create the next generation:

\[ XO((T_1 \cdot TR_1) + ((1 - TR_1) \cdot T_2), (T_3 \cdot TR_2) + ((1 - TR_2) \cdot T_4)) = \]

\[ (((T_1 \cdot TR_1) + ((1 - TR_1) \cdot T_2)) \cdot TR_3) + ((1 - TR_3) \cdot ((T_3 \cdot TR_2) + ((1 - TR_2) \cdot T_4))) \]

Now assume to take two trees of this shape and apply the crossover to generate the offspring of generation 3

And assume to iterate this for hundreds of generations!
Drawback of Geometric Semantic Operators

These operators, by construction, always produce offspring that are larger than their parents, causing an exponential growth in the size of the individuals (proven in [Moraglio et al., 2012]).

This renders them useless in practice.

A solution that has been proposed: “simplification” of the individuals during the evolution. But....
Our Contribution

In:

A New Implementation of Geometric Semantic GP Applied to Predicting Pharmacokinetic Parameters.
L. Vanneschi, M. Castelli, L. Manzoni, S. Silva.
Submitted to EuroGP 2013.
Lecture Notes in Computer Science.

We propose a new implementation of Moraglio’s geometric semantic operators that is efficient and thus allows us to use them (for the first time) on complex real-life applications!
The New Implementation

We create the initial population as in standard GP and we store the trees.
We create all the random trees that we need to produce the next population.
We store the next population like this.

<table>
<thead>
<tr>
<th>Id</th>
<th>Individual</th>
<th>Id</th>
<th>Individual</th>
<th>Id</th>
<th>Operator</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$x_1 + x_2 x_3$</td>
<td>$R_1$</td>
<td>$x_1 + x_2 - 2x_4$</td>
<td>$T_6$</td>
<td>crossover</td>
<td>$(&amp;(T_1), &amp;(T_4), &amp;(R_1))$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$x_3 - x_2 x_4$</td>
<td>$R_2$</td>
<td>$x_2 - x_1$</td>
<td>$T_7$</td>
<td>crossover</td>
<td>$(&amp;(T_4), &amp;(T_5), &amp;(R_2))$</td>
</tr>
<tr>
<td>$T_3$</td>
<td>$x_3 + x_4 - 2x_1$</td>
<td>$R_3$</td>
<td>$x_1 + x_4 - 3x_3$</td>
<td>$T_8$</td>
<td>crossover</td>
<td>$(&amp;(T_3), &amp;(T_5), &amp;(R_3))$</td>
</tr>
<tr>
<td>$T_4$</td>
<td>$x_1 x_3$</td>
<td>$R_4$</td>
<td>$x_2 - x_3 - x_4$</td>
<td>$T_9$</td>
<td>crossover</td>
<td>$(&amp;(T_1), &amp;(T_5), &amp;(R_4))$</td>
</tr>
<tr>
<td>$T_5$</td>
<td>$x_1 - x_3$</td>
<td>$R_5$</td>
<td>$2x_1$</td>
<td>$T_{10}$</td>
<td>crossover</td>
<td>$(&amp;(T_3), &amp;(T_4), &amp;(R_5))$</td>
</tr>
</tbody>
</table>

These are memory references!
<table>
<thead>
<tr>
<th>Id</th>
<th>Individual</th>
<th>7.34</th>
<th>8.62</th>
<th>9.51</th>
<th>4.07</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>$x_1 + x_2 x_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>$x_3 - x_2 x_4$</td>
<td>9.73</td>
<td>4.29</td>
<td>5.26</td>
<td>1.45</td>
</tr>
<tr>
<td>T3</td>
<td>$x_3 + x_4 - 2x_1$</td>
<td>2.92</td>
<td>3.76</td>
<td>5.23</td>
<td>6.24</td>
</tr>
<tr>
<td>T4</td>
<td>$x_1 x_3$</td>
<td>7.28</td>
<td>1.78</td>
<td>3.26</td>
<td>5.74</td>
</tr>
<tr>
<td>T5</td>
<td>$x_1 - x_3$</td>
<td>2.57</td>
<td>4.67</td>
<td>3.22</td>
<td>6.91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Individual</th>
<th>2.64</th>
<th>3.28</th>
<th>5.93</th>
<th>4.29</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>$x_1 + x_2 - 2x_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>$x_2 - x_1$</td>
<td>6.94</td>
<td>7.53</td>
<td>8.53</td>
<td>2.65</td>
</tr>
<tr>
<td>R3</td>
<td>$x_1 + x_4 - 3x_3$</td>
<td>4.84</td>
<td>3.56</td>
<td>2.76</td>
<td>9.76</td>
</tr>
<tr>
<td>R4</td>
<td>$x_2 - x_3 - x_4$</td>
<td>4.37</td>
<td>5.94</td>
<td>2.59</td>
<td>1.85</td>
</tr>
<tr>
<td>R5</td>
<td>$2x_1$</td>
<td>4.67</td>
<td>3.27</td>
<td>2.57</td>
<td>7.47</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Operator</th>
<th>Entry</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T6</td>
<td>crossover</td>
<td>{&amp;(T_1), &amp;(T_4), &amp;(R_1)}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7</td>
<td>crossover</td>
<td>{&amp;(T_4), &amp;(T_5), &amp;(R_2)}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T8</td>
<td>crossover</td>
<td>{&amp;(T_3), &amp;(T_5), &amp;(R_3)}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T9</td>
<td>crossover</td>
<td>{&amp;(T_1), &amp;(T_5), &amp;(R_4)}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T10</td>
<td>crossover</td>
<td>{&amp;(T_3), &amp;(T_4), &amp;(R_5)}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We also store semantics. Storing also the semantics of each individual allows us to calculate the fitness without evaluating the whole expression!! Obtainable directly from here.

Semantics in the next population.
The New Implementation

Before passing to the next generation:

<table>
<thead>
<tr>
<th>Id</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_1)</td>
<td>(x_1 + x_2x_3)</td>
</tr>
<tr>
<td>(T_2)</td>
<td>(x_3 - x_2x_4)</td>
</tr>
<tr>
<td>(T_3)</td>
<td>(x_3 + x_4 - 2x_1)</td>
</tr>
<tr>
<td>(T_4)</td>
<td>(x_1x_3)</td>
</tr>
<tr>
<td>(T_5)</td>
<td>(x_1 - x_3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Individual</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R_1)</td>
<td>(x_1 + x_2 - 2x_4)</td>
</tr>
<tr>
<td>(R_2)</td>
<td>(x_2 - x_1)</td>
</tr>
<tr>
<td>(R_3)</td>
<td>(x_1 + x_4 - 3x_3)</td>
</tr>
<tr>
<td>(R_4)</td>
<td>(x_2 - x_3 - x_4)</td>
</tr>
<tr>
<td>(R_5)</td>
<td>(2x_1)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Operator</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T_6)</td>
<td>crossover</td>
<td>(&amp;(T_1), &amp;(T_4), &amp;(R_1))</td>
</tr>
<tr>
<td>(T_7)</td>
<td>crossover</td>
<td>(&amp;(T_4), &amp;(T_5), &amp;(R_2))</td>
</tr>
<tr>
<td>(T_8)</td>
<td>crossover</td>
<td>(&amp;(T_3), &amp;(T_5), &amp;(R_3))</td>
</tr>
<tr>
<td>(T_9)</td>
<td>crossover</td>
<td>(&amp;(T_1), &amp;(T_5), &amp;(R_4))</td>
</tr>
<tr>
<td>(T_{10})</td>
<td>crossover</td>
<td>(&amp;(T_3), &amp;(T_4), &amp;(R_5))</td>
</tr>
</tbody>
</table>

\(T_2\) is not used anymore: it can be deleted from memory!
The New Implementation

For producing the next generation:

We create another pool of random trees to produce the next population.

We store the next population in another table of pointers.

We eventually simplify removing trees that will never be used.
The New Implementation – Comp. Complexity

We keep in memory:

- (a subset of) the initial population
- a pool of random trees
- a table of memory references

Cost of evolving a population of $n$ individuals for $g$ generations.

At every generation, we need $O(n)$ space to store the new individuals. Thus, we need $O(ng)$ space in total. Since we need to do only $O(1)$ operations for any new individual (since the fitness can be computed using the fitness of the parents), the time complexity is $O(ng)$.

Thus, we have a **linear** complexity with respect to population size and number of generations.
The last step: expression reconstruction

At the end of a run, we have to reconstruct the trees, and those trees are huge, but:

• Usually we just need one individual (the best on training), so we can do the reconstruction only once.

• We can do it offline, at the end of the run, so that we do not slow down the evolution
First experiments ever on real-life applications!

Two regression problems in the field of pharmacokinetics:
• prediction of **human oral bioavailability** (%F)
• prediction of the **protein-plasma binding level** (%PPB)

The %F dataset consists of **359 instances**, where each instance is a vector of **241 features** (molecular descriptor values identifying a drug, followed by the known value of %F for that drug).

The %PPB dataset consists of **131 instances**, where each instance is a vector of **626 features** (molecular descriptor values identifying a drug, followed by the known %PPB for that drug).
Bioavailability results

Expected

Wow!

Statistically significant difference
Plasma Protein Binding Level results

Expected to be good, but this is great!

Statistically significant difference

Wow!

Statistically significant difference
Why this good results on test data?

The geometrical properties of Moraglio’s operators hold independently of the dataset on which individuals are evaluated!
Summary of the contributions

- An efficient implementation of geometric semantic operators, that has allowed us to use them on real-life applications for the first time.

- Excellent results on the studied applications.

- New insights about the generalization ability of geometric semantic operators (without the novel implementation that allowed us to use geometric semantic GP on these complex real-life problems, this interesting property would probably remain unnoticed).
Open issues

The *reconstruction of the expression of the best individual*, even though we do it only once and after the termination of the run, is still an issue:

Individuals after hundreds of generations get so huge that it may be *impossible* to reconstruct their entire expression (even though it is possible to get some information about it, such as the features or primitives it uses...).

Models generated by geometric semantic GP are *black* (or at least "*dark gray*" 😊 ) boxes!

*We are working on this!*
Thank you!